EVERGENTIC · WHITE PAPER

Emisario: the self-managing factory of AI workers

An agent-to-agent (A2A) operations platform that dispatches specialized AI workers, checks its own work, refuses to ship anything unsafe, and gets faster the more it runs.

Platform
Emisario™

Company
Evergentic

Category
Agentic operations / A2A

Audience
Executives · Operators · Engineers · Buyers

Version
1.0 — July 2026

LAYER 1
Executive Summary
The value, the market case, what makes it different.

LAYER 2
Plain English
The lanes idea — no jargon, just how it works.

LAYER 3
Technical Spec
Architecture, the loop, the mesh, the OSS stack.

LAYER 4
Positioning
Why tethered best-of-breed beats a pre-integrated suite.

FOR EXECUTIVES & BUYERS

Executive Summary

Emisario turns a backlog of work into a workforce. It is an agent-to-agent platform: instead of one AI assistant answering one prompt, Emisario runs

a coordinated fleet of specialized AI workers that plan, execute, review each other, and improve — under governance you control.

Most organizations adopt AI as a chat window bolted onto the side of their existing tools. That helps individuals, but it does not change how the work gets done. Emisario changes the unit of work from *"a person prompting a model"* to *"the business dispatching an outcome."* You describe what you want; the platform enriches that into an expert-grade specification, assigns it to the right agents, keeps them on task, has an independent reviewer check the result, and surfaces only the few decisions that genuinely require a human.

> A self-managing factory of AI workers that checks its own work — and gets faster the more it runs.

What it does

Dispatches work

A central orchestrator routes each job to specialized agents across many parallel "lanes," so dozens of tasks progress at once instead of one at a time.

Checks itself

Every result passes an independent reviewer — a separate agent from the one that built it — before anything is considered done. Defects become permanent regression tests.

Ships safely

A fail-closed gate means nothing risky auto-executes. Money, credentials, deletions, access changes and outbound messages always stop for one human approval.

Why it is different

- **It is self-managing.** The orchestrator runs a goal-completion loop — build → test → fix → continue — with stall detection and convergence criteria, so work advances without a human babysitting each step.
- **It is self-improving.** A review flywheel means each run hardens the next: caught defects become tests, verified facts become guardrails, and throughput and reliability climb over time.
- **It is open and portable.** Emisario adopts best-of-breed open-source at each layer and tethers it together with its own connectors — no monolithic suite, no vendor lock-in, and the option to run inference on your own hardware for cost, privacy and control.

- **It federates.** Multiple organizations can run their own Emisario and cooperate through a privacy-preserving mesh, sharing capability without sharing raw data.

Governance first. Emisario is built on a discipline of *verify at the source, never fabricate, never ship on assumption.* New code lands on review branches only and is never auto-merged; the platform already runs behind an edge web-application firewall in detection mode; and a consolidated approval digest puts every sensitive action in front of a human before it happens.

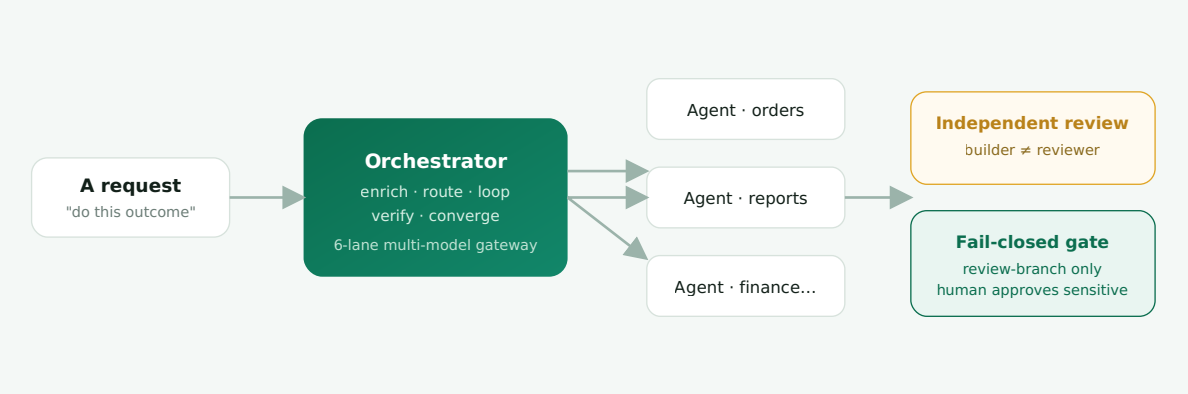

Figure 1. A request enters, the orchestrator enriches and routes it across parallel lanes to specialized agents, an independent reviewer validates the output, and a fail-closed gate governs anything sensitive.

The bottom line

Emisario is not a smarter chatbot. It is operational infrastructure — a governed, self-improving workforce that scales from a single task to a hundred concurrent jobs and, eventually, to a mesh of cooperating organizations. It is built on open foundations you can own, and wrapped in the kind of controls a real business needs before it lets software touch money, data, and customers.

IN PLAIN ENGLISH

Emisario, explained with lanes

Forget the technology for a minute. Picture a highway. Each lane carries one AI worker doing one job. The whole story of Emisario is: how many lanes can you open, and how do you keep the traffic safe?

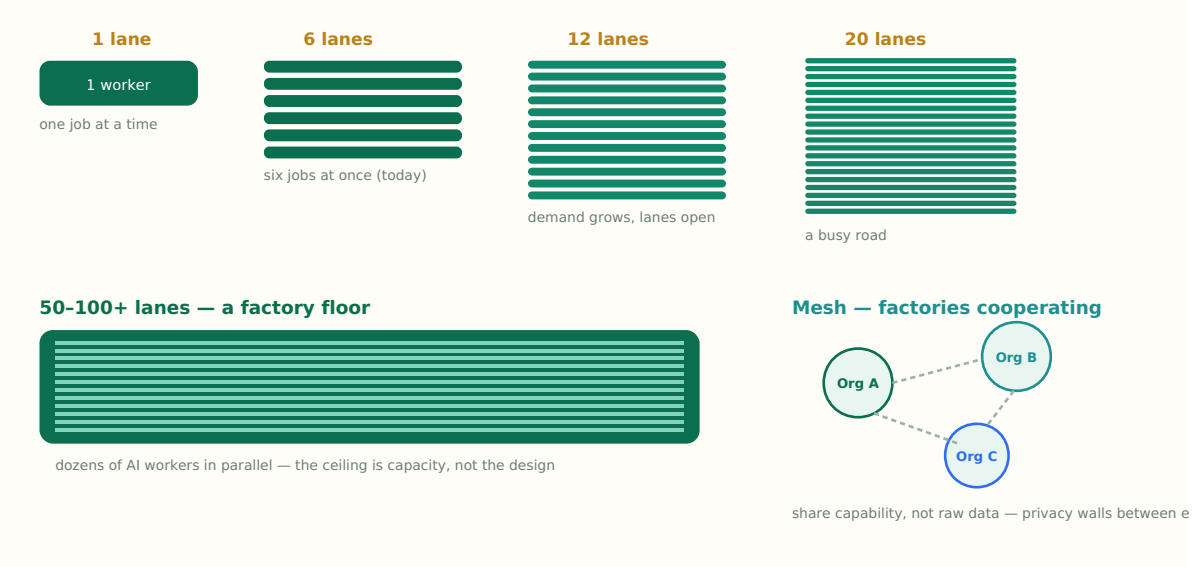

Figure 2. The scale story: **1 → 6 → 12 → 20 → 50–100+ → mesh.** One worker becomes a lane, lanes become a highway, and highways become a cooperating network of organizations.

How the traffic stays sane

A real highway needs more than lanes. It needs a dispatcher, a safety inspector, and an on-ramp that won't let anything dangerous through. Emisario has all three:

The dispatcher

You say what you want in plain words. Emisario quietly rewrites it into a proper, detailed brief and sends it down the right lane — so a two-line request becomes a professional work order.

The inspector

When a worker finishes, a *different* worker inspects the result. If it's wrong, it goes back. Every mistake it catches becomes a permanent check, so the same mistake can't happen twice.

The safety on-ramp

Anything that touches money, passwords, deletions, access, or outgoing messages stops at a gate and waits for a person to say yes. Everything else just flows.

> "Checks its own work" is the inspector lane. "Gets faster the more it runs" is what happens when every caught mistake becomes a lesson the whole factory keeps forever.

Why "gets faster the more it runs"

This is the part that sounds too good to be true, so here is the plain version. Ordinary software does the same thing at the same speed forever. Emisario keeps a memory. Each night it reviews its own recent work, turns every bug it found into a test, and locks in every fact it verified. So next week the same kind of job needs fewer corrections and moves through the lanes faster. The factory isn't just working — it's tuning itself.

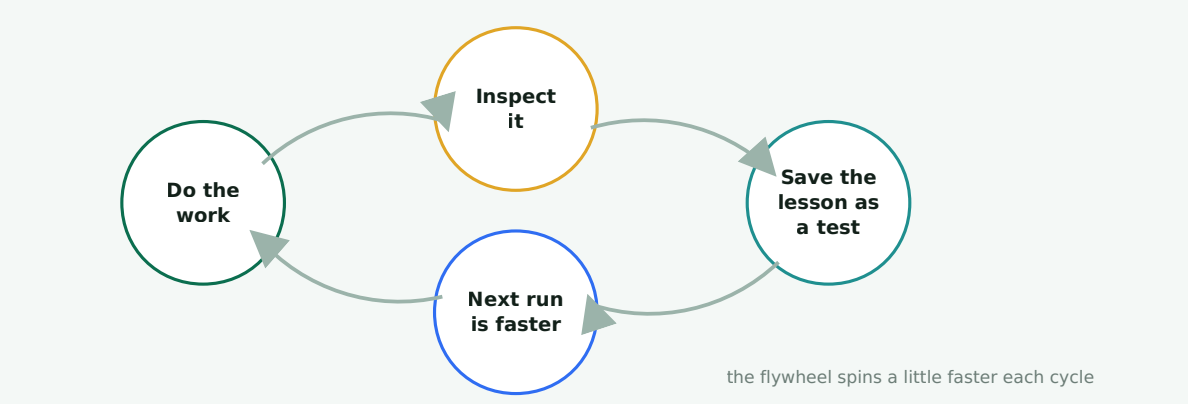

Figure 3. The improvement flywheel, without the jargon: do the work, inspect it, keep the lesson, run faster next time.

TECHNICAL SPECIFICATION

Architecture & internals

Emisario is an A2A (agent-to-agent) platform: a fleet of specialized agents, a looping orchestrator, a multi-provider inference gateway, a knowledge-graph memory, an open-source substrate tethered by purpose-built connectors, and a fail-closed governance layer — deployed behind an edge WAF on Azure Container Apps.

1 · System overview

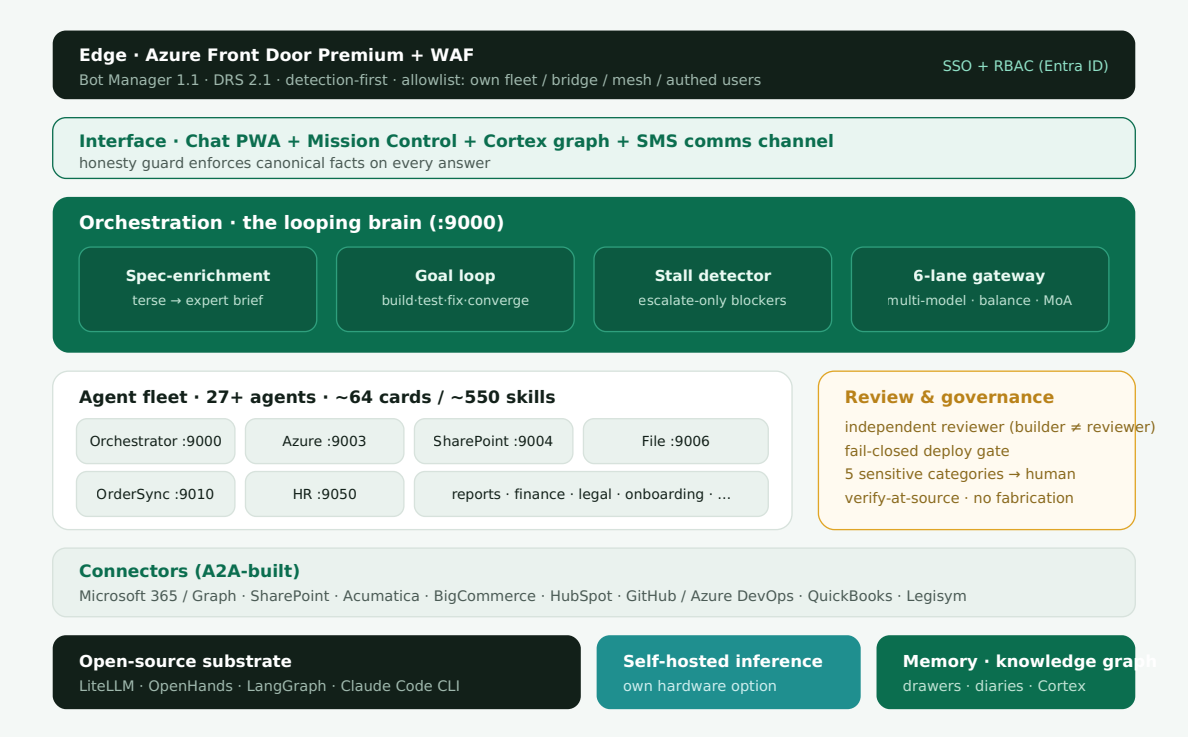

Figure 4. The full stack — edge, interface, the looping orchestrator, the agent fleet, review & governance, A2A connectors, and an open-source substrate with self-hosted inference and knowledge-graph memory.

2 · The loop engine

The orchestrator (:9000) is a *looping expert*. Where a naive agent plans once and declares victory, Emisario runs an outer goal-completion loop with explicit acceptance criteria. Each iteration plans, acts, and then **verifies against criteria** before deciding whether to continue, retry an alternative, escalate a blocker, or finish. A **stall detector** watches for no-progress streaks, repeated failed actions, and human-blocked states (e.g. a missing credential or a permission wall) and surfaces those immediately rather than spinning. A **task budget** caps steps, time and retries so a runaway loop is structurally impossible.

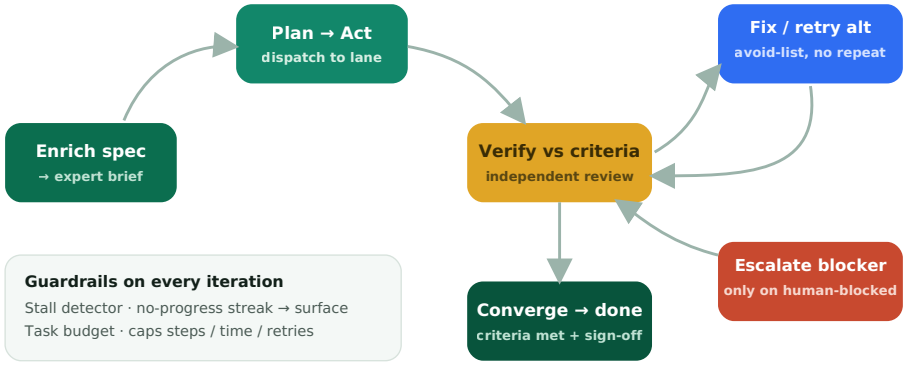

Figure 5. The loop cycle: enrich → plan/act → verify against criteria → fix or escalate → converge. Convergence requires that acceptance criteria are met *and* an independent reviewer signs off; a stall detector and a task budget bound every iteration.

Spec enrichment

Before dispatch, a terse instruction is expanded into a senior-engineer-grade brief. A role router detects the task type and wraps the request in a five-part template — expert role and stakes, a phased approach (understand → diagnose → deliver), an explicit deliverables checklist, hard constraints (*trace the real root cause, do not guess, do not change unrelated functionality, verify at the source*), and a production quality bar. Ten personas cover new builds, codebase audits, debugging, performance, security, QA, review, refactor, teaching, and DevOps/deploy. The effect: "fix the cost tile" becomes a complete debugging brief automatically.

3 · Multi-model gateway & the lanes

Inference is fronted by a config-driven gateway analogous to **LiteLLM**: six routing lanes today, each mapped to a provider account, where adding a lane is *one vault secret plus one config row.* The gateway supports least-loaded balancing, ordered failover across providers (Anthropic, OpenAI, Gemini), and a mixture-of-agents (MoA) fan-out mode. Lanes time-slice, so concurrency is not hard-capped at the lane count — the practical ceiling is aggregate provider rate limits, which is why the roadmap pairs the autonomous loop with multi-account expansion and self-hosted inference.

Proven & targeted throughput

~6 concurrent jobs proven in production today; **50–60** concurrent is the near-term target, gated on the deployed autoloop (so Emisario self-dispatches and self-verifies), multi-provider capacity, and one load test.

Cortex — the live map

An architecture knowledge graph (**427 nodes** / **485 edges** across ~28 tiers, rendered in 3D) that expands with per-agent capabilities toward **~870 nodes** — a self-describing model of the fleet and its skills.

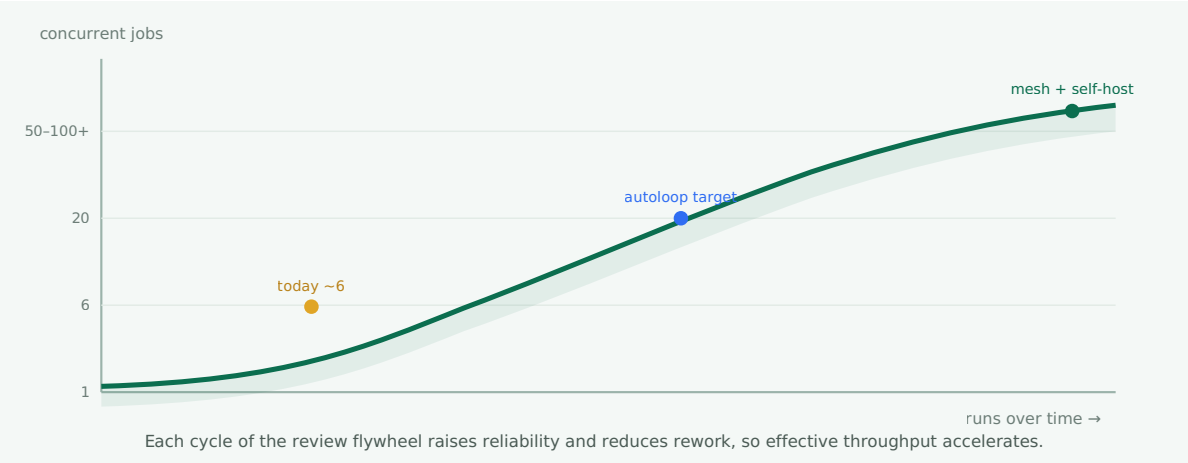

Figure 6. Throughput is not a flat line. As the flywheel converts caught defects into permanent tests, rework falls and effective concurrency climbs — the platform gets faster the more it runs.

4 · The self-improvement flywheel

Emisario improves itself on a schedule (a self-improve pass and a nightly scan). The mechanism is a **separation-of-duties review loop**: the agent that builds a change is never the agent that reviews it. The independent reviewer has repeatedly caught real defects — false-progress bugs, missed edge-cases, classifier gaps — and every catch is **pinned as a regression test**, so the same class of error cannot recur. Verified platform facts are frozen into a machine-checkable **honesty guard** that flags any answer contradicting canonical facts and never silently rewrites output. The net effect is compounding: reliability, coverage, and speed all trend upward as the corpus of tests and verified facts grows.

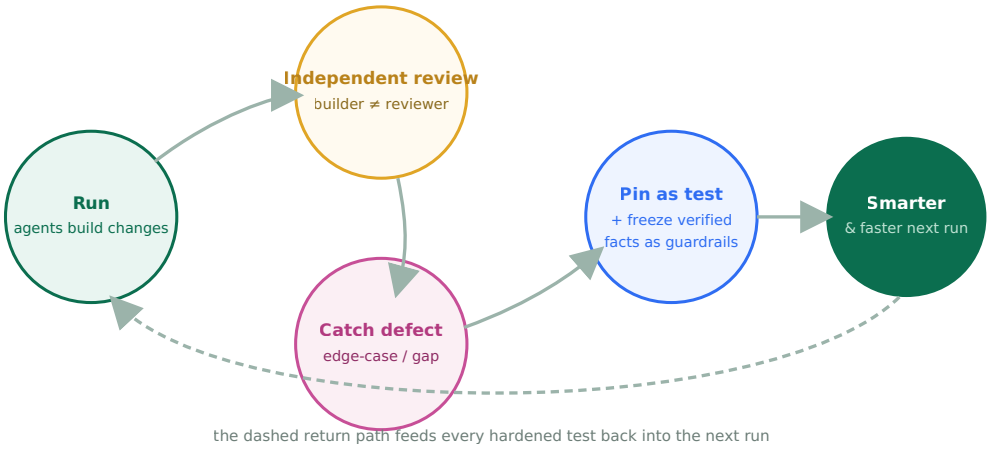

Figure 7. The flywheel: run → independent review → catch defect → pin it as a permanent test and freeze verified facts as guardrails → the next run is smarter and faster. Nothing regresses because every lesson is retained.

5 · Fail-closed deploy gate

Emisario's safety model is **fail-closed**: the default for anything risky is *stop*. Coder agents **edit but do not commit**; work lands on **review branches only and is never auto-merged.** A change must pass its tests and an independent review, and any action falling into one of five sensitive categories is escalated to a human as a single consolidated approval digest. The classifier is fail-safe — it can only *upgrade* an action's sensitivity, never downgrade it — and it was hardened against phrasings that omit an obvious write-verb (e.g. "refund $5k", "store this key"). In production, traffic already flows through an edge WAF running in **detection mode**; the cutover to active prevention is itself gated on a human go/no-go after a clean observation window, with rollback in seconds.

The five sensitive categories

Money · **Credentials** · **Deletion** · **Access / security** · **External comms**

Everything else that is non-destructive proceeds automatically.

Standing rules

Verify at the source · never fabricate · no hollow success · don't clobber a live session (isolated worktrees) · deliver results in chat *and* the system of record · never drop a thread.

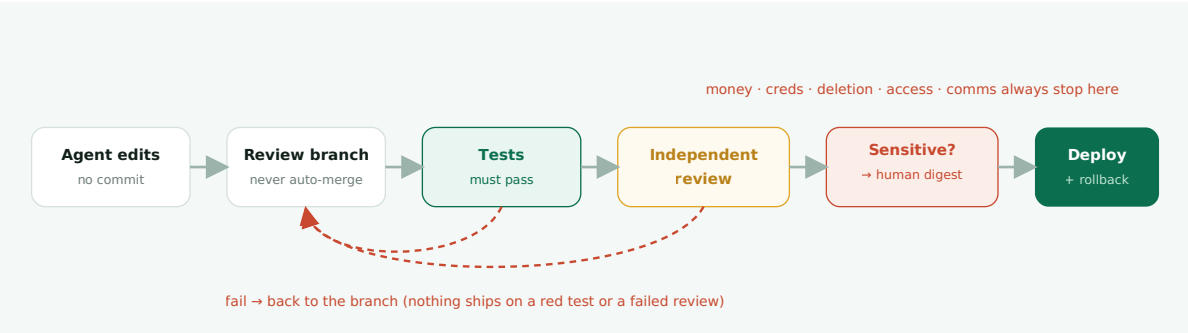

Figure 8. The deploy gate. Work moves left to right only if every check is green; a red test or a failed review sends it back, and any sensitive action diverts to a human approval digest before it can proceed.

6 · Federation mesh

Emisario is multi-tenant at the organization level. Two or more organizations each run their own instance and cooperate through a **federation bridge**. One node acts as primary and peers register as replicas; work crosses the bridge as structured **task envelopes** carrying **AgentCards** (a capability manifest per agent). Crucially, the mesh shares *capability, not raw data*: an **anonymization gate runs fail-closed**, so if scrubbing can't be verified, nothing crosses. Repository mirroring uses short-lived, auto-rotating **GitHub App installation tokens** (least privilege) rather than long-lived secrets, and a key mismatch on the bridge fails closed (a 401, not a leak). This is how a highway of lanes becomes a network of cooperating factories without collapsing anyone's privacy boundary.

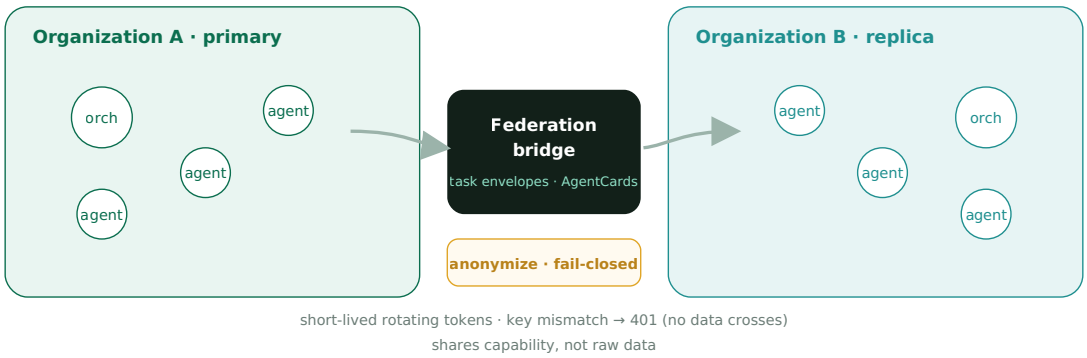

Figure 9. The federation mesh. Organizations peer through a bridge that exchanges task envelopes and capability manifests; a fail-closed anonymization gate and rotating least-privilege tokens ensure capability is shared while raw data stays home.

7 · Open-source substrate & self-hosted inference

Emisario deliberately does not reinvent the layers where open-source already wins. It **adopts** best-of-breed OSS — **LiteLLM** for the multi-provider inference gateway, **OpenHands** for agent execution environments, **LangGraph** for orchestration graphs, and the **Claude Code CLI** for code execution — and **builds** the connective tissue: the A2A connectors, the loop and review policies, the governance gate, and the federation bridge. Because inference sits behind a provider-agnostic gateway, Emisario can add a hosted provider lane or a **self-hosted inference** endpoint with the same one-secret-plus-one-row change — giving operators a lever on cost, latency, data residency, and privacy that a closed suite cannot offer.

LAYER	APPROACH	WHAT EMISARIO USES / BUILDS
Inference gateway	Adopt	LiteLLM-style multi-provider routing + balance + failover; self-hosted endpoints as first-class lanes
Agent execution	Adopt	OpenHands execution environments; Claude Code CLI for code tasks
Orchestration graphs	Adopt	LangGraph for stateful agent graphs
Loop & review policy	Build	Goal loop, stall detector, task budget, separation-of-duties review loop
Connectors	Build	M365/Graph, SharePoint, Acumatica, BigCommerce, HubSpot, GitHub/ADO, QuickBooks, Legisym
Governance	Build	Fail-closed gate, sensitive-action classifier, honesty guard, edge WAF
Memory	Build	Knowledge-graph drawers + diaries + the Cortex architecture graph
Federation	Build	Bridge, task envelopes, AgentCards, fail-closed anonymization

The trifecta / own-platform trajectory

The same philosophy extends to the applications Emisario runs. Rather than remain locked to a rigid pre-integrated ERP suite whose customization surface resists automation, Evergentic's roadmap includes a **metadata-driven core** (PostgreSQL + FastAPI + React) validated by a **mirror-to-zero-variance** method against the incumbent before any cutover, with usage-driven

lean scoping, reporting held on a separate surface, and 365 SSO. Best-of-breed components tethered by A2A connectors replace a monolith — on the operator's terms and timeline.

POSITIONING & USE CASES

Why tethered best-of-breed beats a pre-integrated suite

The market's default answer to "we need AI in our operations" is a pre-integrated suite: one vendor, one bundle, one roadmap. It is convenient on day one and constraining on every day after. Emisario takes the opposite bet — adopt the best component at each layer, own the connectors that bind them, and keep the governance in your hands.

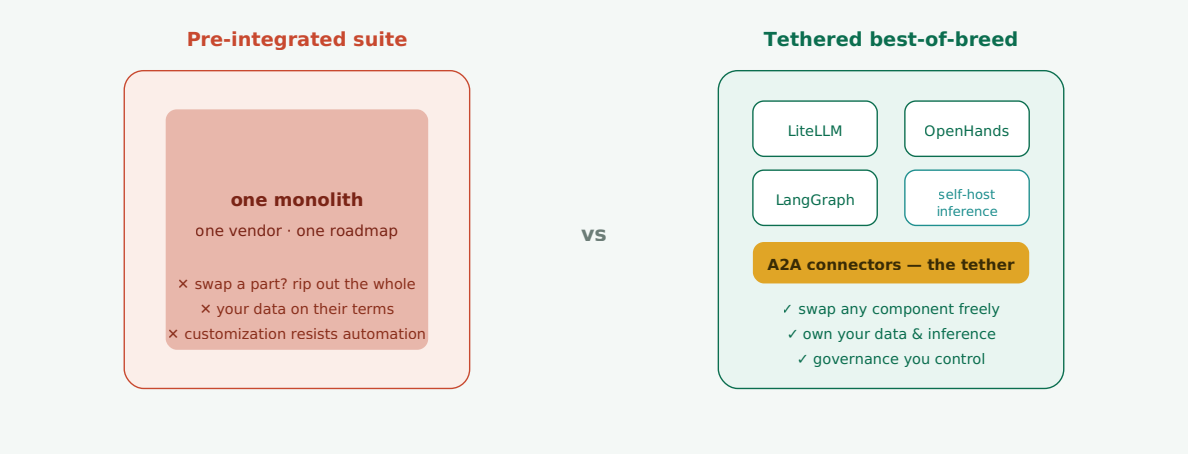

Figure 10. A suite bundles everything into one replaceable-only-as-a-whole monolith. Emisario keeps each layer independently best-in-class and binds them with A2A connectors — so you can swap, self-host, and govern on your own terms.

Positioning in one line

> Emisario is the governed workforce layer that sits on open foundations you own — not another silo you rent.

No lock-in

Every layer is swappable. Change model providers, execution backends, or connectors without re-platforming. Your leverage stays with you.

Own the economics

Route to hosted models or self-hosted inference through the same gateway. Tune cost, latency, and data residency instead of accepting a bundle price.

Auditable by design

Review branches, independent review, an honesty guard, and a consolidated approval digest make every consequential action inspectable — a prerequisite for regulated operations.

Compounding returns

The flywheel means the platform you buy this quarter is slower than the one you'll have next quarter, at no extra integration cost.

Scales with demand

From one lane to a hundred to a federated mesh — the same architecture, no forklift upgrade between stages.

Federation-ready

Cooperate with partner organizations without merging data estates. Share capability across a privacy-preserving mesh.

Where it earns its keep

USE CASE	WHAT EMISARIO DOES	WHY IT'S SAFE
Order operations	Ingests POs, syncs to ERP, catches overrides and leaks, recovers mis-routed records	Corrective writes gated through the approval digest
Reporting & reconciliation	Audits reports, reconciles against source of record, flags drift	Read-first; access-gated; no silent changes
Accounting prep	Cash / credit-card reconciliation analysis, AP bill preparation	Money posting stays human-gated — prepares, never posts
E-commerce catalog	Keeps storefront descriptions and SKUs in sync with the ERP	Changes staged and verified before sync
Onboarding / offboarding	Coordinates identity, groups, mailboxes, and handoffs	Identity & access actions require a human yes
ERP customization	Drives customization via API with windowed publish / smoke / rollback	Sandbox-first; publish after-hours with an owner go
Multi-org collaboration	Hands packaged work to a partner org across the mesh	Fail-closed anonymization; capability, not raw data

Objections, answered

"Isn't a bundle simpler?"

Simpler to buy, harder to live with. The first time you need to replace one weak component, a suite forces an all-or-nothing migration. Tethered best-of-breed turns that into a one-connector change.

"Can I trust agents with real work?"

Only within a fail-closed gate. Nothing touching money, credentials, deletions, access, or outbound messages executes without a human. The platform is designed to prove its work, not assert it.

"What about vendor risk?"

The substrate is open-source and the inference is provider-agnostic — including self-hosted. If any single vendor changes terms, you re-route, you don't re-platform.

"Will it stall or run away?"

Neither. A stall detector surfaces human-blocked states immediately, and a task budget caps steps, time, and retries so a loop can't spiral.

Naming note. Throughout Evergentic materials, **Emisario** is the platform and **A2A** is used only as the technology descriptor — agent-to-agent. The two are not interchangeable: A2A is how Emisario works; Emisario is what you adopt.

REFERENCE

Diagram index

This paper contains ten visuals designed to be read on their own: **Fig 1** Emisario at a glance · **Fig 2** the scale/lanes progression · **Fig 3** the improvement loop (plain) · **Fig 4** the full architecture stack · **Fig 5** the loop cycle · **Fig 6** throughput over time · **Fig 7** the self-improvement flywheel · **Fig 8** the fail-closed deploy gate · **Fig 9** the federation mesh · **Fig 10** tethered best-of-breed vs a suite.